REMARKS/ARGUMENTS

Claim 4 has been amended to be dependent on claim 1. In addition, minor typographical errors have been made. No substantive claim amendment has been made. As such, it is requested that the Examiner enter the claim amendment.

A proposed drawing of Fig. 2 labeled as "Replacement Sheet" has been submitted. Accordingly, it is respectfully requested that the Examiner enter the amendment and withdraw all objections to the claims under 35 U.S.C. 112, second paragraph and objections to the drawings.

The Examiner has maintained his objection of the specification and the rejection of claims under 35 U.S.C. §102 and 35 U.S.C. §103(a) as being anticipated by ColdFusion 4.0 software product documents. The objection to the specification and claim rejection are fully traversed below.

Objection to the Specification

In the Final Office Action, the Examiner has maintained the objection to the specification because the Examiner believes that every occurrence of the trademark name Java must be used as a noun and be accompanied with generic terminology.

The Applicant has painstakingly capitalized trademark names in an attempt to respect the proprietary nature of the marks and to make every effort to prevent their use in a manner that might adversely affect their validity as trademarks, pursuant to guidelines set forth in MPEP § 608.01(v). It is respectfully submitted that the undersigned earnestly believes that there is no legal requirement that would suggest that every instance of trademark language must be accompanied by generic terminology and/or be used as a noun.

Furthermore, it is respectfully submitted the some programming languages or specific software products are well known to those skilled in the art by a trademark name (e.g., Java programming language). As such, it is NOT necessary to accompany trademark names such as the Java programming language with generic terminology. In fact, it may be inappropriate to use generic terminology if a programming language or product is primarily known to those skilled in the art by its trademark name (e.g., Java

programming language). (Please see, for example, the Board of Appeals decision to the effect that a product on the market should be known generally to those skilled in the art or it be necessary to use the trade mark to identify the product; *Ex parte Frederick and Waterfall*, 75 USPQ 298 (Bd. Pat. App. & Int. 1947).) Nevertheless, solely in order to expedite prosecution, the trademark name Java has been accompanied by generic terminology because of the Examiner's insistence. Clearly, the Applicant is NOT required to accompany every occurrence of the trademark name Java with generic terminology. Still furthermore, the Applicant is using its own trademark name. As such, the Applicant should not be required to use it in a way that the Examiner deems appropriate.

The Applicant has painstakingly made every effort to expedite prosecution of the present application and respectfully requests that the Examiner withdraw all rejections to the specification in order to avoid further delay in prosecution of the case on it merits.

Rejection of claims

In the Office Action, the Examiner has rejected claims 1-4, 7, 8, 13-15, 17 and 18 under 35 U.S.C. §102(a) as being anticipated by ColdFusion 4.0 software product documents.

(a) The cited art does not teach translation of the page to an executable programming code

To support this rejection, the Examiner has asserted that the ColdFusion 4.0 software product, developing web applications with ColdFusion ("*CF Web*"), teaches a translator suitable for translating an action tag from a mark-up language to an executable programming code that is executed at runtime to perform actions intended by the action tags (Office Action, page 7, paragraph 10, citing CF web, page 7, first three paragraphs). Contrary to the Examiner's assertion, *CF Web* describes a conventional environment where a browser interprets an HTML page and generates output directly from the interpretation of the HTML page. It should be noted that the ColdFusion only processes a page and then outputs the HTML to the web browser that interprets it. Clearly, *CF Web* does not teach translation of the page to an executable programming code (e.g., Java Programming language) that is, in turn, executed (e.g.,

interpreted by a virtual machine) at runtime to generate output. Accordingly, the Examiner's rejection is improper and should be withdrawn.

(b) The cited art does not teach a TagExtraInfo

Furthermore, it is respectfully submitted that the "Ancestor data access" described in *CF Advanced* fails to teach a TagExtraInfo object for each action tag in a page. It should be noted that the TagExtraInfo object provides a method that is accessed by a translator at translation time. The method returns, at translation time, information that includes a list of available scripting variables, and a variable type and scope associated with each scripting variable that is defined or modified by its associated action tag. As will be appreciated, the method allows the translator, at translation time, to use the information provided by the method to generate code that, when executed at runtime, will assign each of the scripting variables with appropriate runtime values with respect to the type and scope of each of the scripting variables.

Clearly, the "Ancestor data access" described in *CF Advanced* fails to teach these features. This is evident because even from a very broad perspective a function that returns "ancestor data" merely provides information about an ancestor. This ancestor information can be used to determine nesting (i.e., what call you). However, there is no requirement for an ancestral relationship between two variables in the list of available scripting variables. As such, a function that merely provides an ancestor data access is inadequate. Accordingly, even from a very broad perspective, the "ancestor data access" of *CF Advanced* cannot be used to provide a list of available scripting variables, and a variable type and scope associated with each scripting variable because it only provides information about an ancestor.

Moreover, the "ancestor data access" *CF Advanced* cannot be used by a translator that generates executable code. This is evident because there is no translator or translation of executable code in the *CF Advanced* environment. It should also be noted that there is no motivation or suggestion to use the "ancestor data access," to provide the list of available scripting variables, notwithstanding the fact that such a combination may be technologically absurd. It should also be noted that the Examiner has not even addressed the feature of a TagExtraInfo object for each action tag in a page. Instead, the Examiner has merely noted that a function (getbaseTaglist()

) can be called to return a comma-delimited list of uppercase ancestors tag names (Office Action, page 8, citing the fifth line of the code sample).

Accordingly, the Examiner's rejection is improper and should be withdrawn for additional reasons.

(c) The cited art does not teach a pageContext object

Still further, it is respectfully submitted that the cited art does not teach a pageContext object for a page. The pageContext provides a runtime mapping of at least one scripting variable in the list of variables. As noted above, the list of variables were provided (returned) by the method of TagExtraInfo object at translation time. In the Office Action, the Examiner has asserted that these features are taught by CF Web, pages 16-17. Contrary to the Examiner's assertion, CF Web merely teaches creating and using variables in the Coldfusion software environment. Clearly, this tutorial does not teach providing a runtime mapping of at least one scripting variable in the list of variables that has been generated at translation time. It should also be noted that the Examiner seems to have made a general allegation without providing a specific reference that "using Application and Session variables," "Using CGI Environment Variables," and "creating HTTP cookie variables" teach these features. This general allegation does not even address the claimed features because the Applicant has not broadly claimed using a scripting variable. Nevertheless, the Examiner seems to have merely selected various sections of the art that merely make a reference to the word "variable." Clearly, the Examiner has not addressed these features because, aside form the word "variable" in the title of the cited sections, the Examiner has not provided any factual basis to support the rejection.

Accordingly, the Examiner's rejection is improper and should be withdrawn for yet additional reasons.

(d) The cited art does not teach a tag handler

The rejection is improper for yet another additional reason because the cited art does not teach a tag handler that creates at runtime one or more objects than the page requires. It should be noted that the tag handler further operates to store the one or

more created objects into the pageContext object. This allows the one or more objects to be retrieved at runtime when the generated code (generated at translation time) is executed. The one or more objects are assigned at runtime to the variables in the list of scripting variables that is returned by the method at translation time.

In the Final Office Action, the Examiner has asserted that these features are taught by the "Dynamic Parameter" and " Expression" examples (Final Office Action, page 9). Contrary to the Examiner's assertion, these examples merely demonstrate using typeless variables in ColdFusion. When a typeless variable is used, the variable type does not need to identified. Accordingly, a variable can be based on a column name or an expression (CF Web, page 17). The typeless variables, however, do not create at runtime one or more objects that the page requires. Moreover, the typeless variables do not store the one or more created objects into another object (e.g., the pageContext object). Clearly, using a dynamic variable does not address the claimed features of: a tag handler that creates at runtime one or more objects that the page requires and stores the one or more created objects into a pageContext object.

It should also be noted that the Examiner has made general allegations that various sections of the cited art somehow teach these features. This general allegation does not even address the claimed features because the Applicant has not broadly claimed using a scripting variable. Nevertheless, the Examiner seems to have merely selected various sections of the art that merely make a reference to the word "variable." Clearly, the Examiner has not addressed these features because, aside form the word variable in the title of the cited sections, the Examiner has not provided any factual basis to support the rejection.

Accordingly, the Examiner's rejection is improper and should be withdrawn for still additional reasons.

(e) The Rejection is improper for several other reasons

The claimed invention provides a tag library. It should be noted that pageContext object for the page includes a runtime mapping of at least one scripting variable in the list of available scripting variables to a runtime value that is represented or can be represented in the tag library. The Examiner has not addressed these features.

Instead, the Examiner has merely asserted that providing a tag library as a collection of tags is known (Final Office Action, page 7).

The Examiner has also failed to address allowing a translator at translation time to use the information provided by a method to generate code that when executed at runtime will assign each of the scripting variables with appropriate runtime values with respect to the type and scope of each of the scripting variables. Again, the Examiner has not addressed these features because the Examiner has not provided any factual basis to support the rejection.

The Examiner has also failed to address allowing one or more objects to be retrieved at runtime when the generated code is executed, wherein the one or more objects are assigned at runtime to each of the scripting variables in the list of scripting variables that is returned by a method at translation time. The Examiner has not addressed these features because the Examiner has not provided any factual basis to support the rejection.

Summary

Based on the foregoing, it is earnestly believed that it is clearly evident that the Examiner's rejection is improper for several reasons. Moreover, the cited art fails to teach many of the recited features. These features are associated with an environment where mark-up language (e.g., HTML) is translated into executable code in another programming language (e.g., Java programming language) which is then executed at runtime. Also, these claimed features provide automatic synchronization of scripting variables between a page and a tag library. The cited art cannot possibly teach these features because it pertains to a conventional environment where HTML code is processed and interpreted. In fact, there is no need for automatic synchronization synchronization of scripting variables between a page and a tag library in this conventional environment because the markup language is directly interpreted (i.e., there is no translation).

Clearly, the cited art cannot possibly teach or suggest the combination of the claimed features as it is grossly deficient in even teaching a general environment where mark-up language code (e.g., HTML) is translated into executable code in a

programming language (e.g., Java programming language) before it is executed at runtime.

Based on the foregoing, it is respectfully submitted that claim 1 and its dependent claims are patentable for at least these reasons. Furthermore, other independent claims recite similar features as those recited in claim 1. Therefore, it is respectfully submitted all pending claims are patentably distinct over the cited art of record. Additional limitations recited in the independent claims or the dependent claims are not further discussed as the above-discussed limitations are clearly sufficient to distinguish the claimed invention from the cited art. Accordingly, it is respectfully requested that the Examiner withdraw all rejections.

Applicant believes that all pending claims are allowable and respectfully requests a Notice of Allowance for this application from the Examiner. Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

If there are any issues remaining which the Examiner believes could be resolved through either a Supplemental Response or an Examiner's Amendment, the Examiner is respectfully requested to contact the undersigned attorney at the telephone number listed below.

Applicants hereby petition for an extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 500388 (Order No. SUN1P254).

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

R. Mahboubian
Reg. No. 44,890

P.O. Box 778
Berkeley, CA 94704-0778
(650) 961-8300